DOCUMENT RESUME

ED 142 169 IR 004 827

AUTHOR Neches, Robert
TITLE Intelligent Educational Dialogue Systems. Technical Report No. 7701.
INSTITUTION California Univ., San Diego. Center for Human Information Processing.
SPONS AGENCY Office of Naval Research, Arlington, Va. Personnel and Training Research Programs Office.
PUB DATE Mar 77
CONTRACT N00014-76-C-0628
NOTE 42p.; Best copy available

EDRS PRICE MF-$0.83 HC-$2.06 Plus Postage.
DESCRIPTORS Artificial Intelligence; *Computer Programs; Input Output; Programing Languages; *Teaching Methods; Teaching Techniques; *Tutoring; Tutors

ABSTRACT

This paper describes a hypothetical computer program capable of emulating the behavior of a human tutor. Such a program must, ideally, display a number of characteristics. It must be mixed-initiative, meaning that either student or tutor may initiate an interaction. It must be capable of understanding natural language input and generating natural language output. It must be highly knowledgeable about its subject matter, and generative in order to be able to answer unanticipated questions. Finally, the system must be flexible enough to detect the needs of a student and alter its teaching strategy so as to best meet those needs. The first part of the paper discusses what the major parts of the system would be, the functions each part would carry out, and the means by which they would interact. The goal is to develop a general system that can be used with a number of different teaching strategies. The second part of the paper analyzes the Socratic teaching method in order to determine the requirements for implementation within the systems. (Author)

****
* Documents acquired by ERIC include many informal unpublished *
* materials not available from other sources. ERIC makes every effort *
* to obtain the best copy available. Nevertheless, items of marginal *
* reproducibility are often encountered and this affects the quality *
* of the microfiche and hardcopy reproductions ERIC makes available *
* via the ERIC Document Reproduction Service (EDRS). EDRS is not *
* responsible for the quality of the original document. Reproductions *
* supplied by EDRS are the best that can be made from the original. *
****

ED142169

REPORT No. 7701
MARCH 1977

U S DEPARTMENT OF HEALTH, EDUCATION & WELFARE
NATIONAL INSTITUTE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

IR

BEST COPY AVAILABLE

Robert Neches

INTELLIGENT EDUCATIONAL DIALOGUE SYSTEMS

BEST COPY AVAILABLE

IR 004 927

UNIVERSITY OF CALIFORNIA, SAN DIEGO

CENTER FOR HUMAN INFORMATION PROCESSING
LA JOLLA, CALIFORNIA 92093

This research was supported by the Advanced Research Projects Agency and the Office of Naval Research, Personnel and Training Research Programs and was monitored by ONR under Contract N00014-76-C-0628, NR 154-387, under terms of ARPA Order No. 2284. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency, of the Office of Naval Research, or the United States Government.

Approved for public release; distribution unlimited. Reproduction in whole or part is permitted for any purpose of the United States Government.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 7701	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Intelligent Educational Dialogue Systems		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Robert Neches		8. CONTRACT OR GRANT NUMBER(s) N00014-76-C-0628
9. PERFORMING ORGANIZATION NAME AND ADDRESS Center for Human Information Processing University of California, San Diego La Jolla, CA 92093		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR 154-387
11. CONTROLLING OFFICE NAME AND ADDRESS Personnel and Training Research Programs Office of Naval Research (Code 458) Arlington, VA 22217		12. REPORT DATE March 1977 13. NUMBER OF PAGES 26
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Intelligent tutor, dialogue system, teaching strategies, question-answering, model of the student, Socratic method, instructional system.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

(Over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

This paper describes an *"Intelligent Educational Dialogue System"*, a hypothetical computer program capable of emulating the behavior of a human tutor. Such a program must, ideally, display a number of characteristics. It must be ***mixed-initiative***, meaning that either student or tutor may initiate an interaction. It must be capable of understanding natural language input and generating natural language output. It must be highly ***knowledgeable*** about its subject matter, and ***generative*** in order to be able to answer unanticipated questions. Finally, the system must be ***flexible*** enough to detect the needs of a student and alter its teaching strategy so as to best meet those needs.

The first part of the paper discusses what the major parts of the system would be, the functions each part would carry out, and the means by which they interact. The goal is to develop a general system that can be used with a number of different teaching strategies. The second part of the paper analyzes the Socratic teaching method in order to determine the requirements for implementation within the system.

Intelligent Educational Dialogue Systems

Robert Neches

Center for Human Information Processing
University of California, San Diego
La Jolla, California 92093

Report No. 7701
March 1977

The views and conclusions contained in this document are those of the author and do not necessarily reflect the policy of any agency of the United States Government. The research was supported by the Advanced Research Projects Agency and the Office of Naval Research and was monitored by ONR under Contract N00014-76-C-0628. The report is approved for public release; distribution unlimited. Reproduction in whole or in part is permitted for any purpose of the United States Government.

Robert Neches is now at the Psychology Department, Carnegie-Mellon University, Pittsburgh, PA 15213.

PART I -- INTELLIGENT EDUCATIONAL DIALOGUE SYSTEM COMPONENTS

OVERVIEW

Part I describes the components of the system in sufficient detail to motivate the analysis of a Socratic teaching strategy given in Part II. Certain aspects of the system's structure, particularly with respect to issues of control structure and inter-process communications are glossed over here. These issues are considered more thoroughly elsewhere (Neches, 1976); a brief summary is necessary to motivate this section.

I make an effort to expedite studies of tutoring by providing a meta-system in which tutoring strategies are easy to specify; this provides a formalism for studying and comparing different teaching strategies in a common conceptual framework. Models of particular teaching strategies can then be built and studied, as Part II demonstrates.

These goals affect several facets of the system's organization. The system is divided into a number of separate components, which might best be thought of as independent, very large, "Knowledge Sources". This organization is motivated primarily by considerations of pedagogy and methodology, rather than psychology. Decomposing the system into components (each of which represents a major issue in Intelligent CAI and tutorial modelling) simplifies discussions of internal system operations since the processes have been made explicit. The decomposition simplifies presentation of interacting parallel processes, since each component can be treated simply as a "black box" when not itself under consideration. Another virtue of this organization is that it encourages dealing with each of the major issues, because a particular model must further specify the general components provided in order to function.

THE COMPONENTS OF THE SYSTEM

The system is viewed as having seven major components:

(1) an *Input/Output Handler* -- translates the student's language into the system's and vice versa;

(2) *Topic Knowledge Representation (TKR)* -- holds all information about the subject to be taught;

(3) *Model of the Student (MoS)* -- holds student information used by processes such as Planner and I/O Handler;

(4) *Planner* -- "tunes" the system to the student's particular needs, determining the topics presented, their sequence and manner of presentation, and level of detail;

(5) *Session History* -- records the interaction between student and system, as well as intermediate internal states of the other component processes;

(6) *Pragmatics* -- makes real world information available to the system;

(7) *Question-Answering (Q/A)* -- guides information retrieval from the other components (e.g., TKR) by acting as a central control.

The last three components are included here for the sake of completeness. Since they are not directly relevant to the issues discussed in Part II. This section's scope will be confined to the first four components.

THE INPUT/OUTPUT HANDLER

The I/O Handler's first duty is receiving communications from the student, determining their meaning/intent, and representing them in an internal language. This task uses the Model of the Student, Pragmatics, and Topic Knowledge Representation (to obtain information about vocabulary, plausibility of its interpretations, etc.), as well as Session History (to interpret input with respect to the current context).

The I/O Handler's second duty is generating output to the student based on an internal request generated by one or more of the other processes. The request describes the kind of information to transmit (e.g., whether or not an answer was correct) and controls variables such as level of sophistication, but is not itself an English-level statement. Thus, no other process is concerned with language processing, although the I/O Handler calls on Planner, Pragmatics, Topic Knowledge Representation, and Model of the Student to obtain information used in constructing the actual text.

Though much more general, this organization is similar in many respects to the ISI Information Automation Tutor (Rothenberg, 1975), which has pre-stored information varying along dimensions of "verbosity" and "sophistication". Messages are constructed for a particular student by using a "User Profile" to select locations along the two dimensions.

TOPIC KNOWLEDGE REPRESENTATION

The system was designed to teach two sorting algorithms from computer science: the "straight selection sort" and the "binary tree sort" (Knuth, 1973, pp. 139-141 & 428).

In tutorial protocols, four kinds of knowledge about algorithms can be distinguished. (The examples come from actual tutorials. They are presented in boxes.) The four kinds are:

(1) *Structure* -- the algorithm viewed in terms of its sub-algorithms and the flow of control between them. Box 1 shows a simple example. Structure knowledge is multi-leveled, as Box 2 illustrates.

(2) *Motivations* -- reasons both for the purpose and the design of algorithms. This subsumes real world examples (Box 3), interactions between parts (Box 4), and weighing of advantages (Box 5).

(3) *Properties* -- the basic properties, advantages, and disadvantages of algorithms; Box 6 is an example.

(4) *Effects* -- i.e., "what happens if...". Box 7 shows a tutor explaining a proposal's effects.

Insert Boxes 1-7 about here

How can these different kinds of knowledge be represented? The answer proposed here combines procedural and declarative representations.

An algorithm's formal structure is expressible in a ***structure diagram***, a formalism for representing both organization and flow of control in computer programs. The underlying principles are those of structured programming (Dahl, Djikstra, & Hoare, 1972). Structure diagrams represent programs as a tree structure; depth in the tree indicates decomposition into simpler functions, left-right order of branches indicates sequence of processing.

Figure 1 shows the "build tree" algorithm's diagram [1] encoded in a "***structure network***", which uses a modified version of the LNR memory representation (Norman, Rumelhart, and the LNR Research Group, 1975). Although nodes usually consist only of a name and pointers to other nodes, the modifications add "***structure nodes***" (denoted in figures by double lines) having a broader range of associated information: a name, a type (choice, action, or repetition), and a description of its contents. Structure nodes represent either "***tasks***", processes of unspecified method (e.g., sorting), or "***algorithms***", processes of specified method (e.g., binary tree sorting). It is sometimes useful to think of algorithms as tasks for which only one method is specified.

Insert Figure 1 about here

Task structure nodes have a "METHOD" relation to a token of the n-ary predicate "OR", which specifies alternative algorithms for accomplishing the task. Algorithms have a "DO" link to either a structure node or a token of the n-ary predicate "DO-IN-ORDER", which indicates the sequence in which components of the

algorithm are carried out. The DO-IN-ORDER and OR predicates are illustrated in Figure 2.

Insert Figure 2 about here

Figure 2 also illustrates the "HAS-PROP" predicate, which asserts that some subject S HAS-PROPerty P. The token in Figure 2 asserts that the binary tree sort HAS-PROP that its speed is "fast". Use of the HAS-PROP predicate allows assertions about properties to become arguments of other predicates, while still allowing the effect of maintaining a property list (since a structure node can be searched for all associated HAS-PROP tokens).

Motivations will be represented propositionally in a modified story grammar (Rumelhart, 1975), simplified by using assumptions such as that an algorithm is selected either because its properties meet a goal's requirement or because its alternatives don't. This is illustrated by the "WHEN", "REQUIRE", and "USE" predicates of Figure 2. These state the motivation, "When sorting requires speed, use the binary sort because it's fast."

Process Knowledge requires the ability to simulate and examine the execution of an algorithm. Scragg's thesis on answering process questions (Scragg, 1975) suggests an attractive approach, representing processes as procedures. Questions are answered by examining the procedures' code, and/or executing the code under special supervision. The structure network representation can provide all of the information needed in Scragg's approach.

The representational scheme just described captures the four kinds of knowledge about algorithms to at least a first degree of approximation. It is fairly easy to see why this knowledge is useful to both tutor and student. The different knowledge types make for an extremely rich node network, in which concepts are firmly anchored by many links to nodes for other concepts. The latter three knowledge types, constituting knowledge *about* an algorithm as opposed to knowledge *of* it, allow a student who has forgotten the structure to recompute it. That is, knowledge from any three of the types often provides sufficient constraints on the situation to permit inferencing knowledge of the fourth type in relative safety.

These arguments for the knowledge types' utility are best understood in the context of Norman's "web learning" concept (Norman, 1973, 1974). Norman argues that a network of rich interconnections is essential for effective learning and retention of the kind of concepts we wish to teach. The teaching process, he believes, must consist of many passes through a network of knowledge to be taught. This permits a concept to be touched on many times, in many different contexts. Thus, a student benefits from multiple exposures to a concept, as well as strong anchoring to other, already familiar concepts. Since the knowledge representation presented here admits both hierarchical and non-hierarchical relationships between concepts, it allows the capability to take these multiple passes through the knowledge network at varying levels of detail and sophistication. This is a key to the web learning approach: unlike

the contrasting approach of "sequential learning", web learning ensures that the student always maintains a coherent knowledge structure. The structure at times may be incomplete, but it will never be unstable.

MODEL OF THE STUDENT (MoS)

Several examples in this paper illustrate a responsive system's need for a database of information about the student's current knowledge state and general propensities.

Direct and inferential data about students is gathered by examining their input via the I/O Handler, Topic Knowledge Representation, and Session History. In addition, analyses of its data performed elsewhere can feed back to MoS. The Model of the Student must supply its information to several other processes, working particularly closely with the I/O Handler and Planner. The I/O Handler uses the information both in understanding student input and in generating system output. Planner uses the information in selecting teaching rules.

Information must be collected when available rather than when requested, requiring the ability to anticipate what information will be needed. This ability is embodied in processes called *Knowledge State Conditions* (KSCs). These can be thought of as procedures processed partly by the Planner and partly by MoS. The definition of a Knowledge State Condition contains:

(1) an *information collecter* -- instructions specifying what information is needed and how it's to be obtained;

(2) an *analysis section* -- an algorithm for determining how well the condition has been met by examining the information collector's data;

(3) a *weighter* -- an algorithm for computing the condition's importance as a function of the system's current state.

In this scheme, Planner processes the weighting information, MoS the information collectors. The analysis section is shared (Planner invokes, MoS processes). Each section of a KSC runs separately. Information collectors function like the demons of Selfridge and Neisser (1960), invoked at any time by student input or internal system states. Analysis and weighting are top-down functions, invoked only by direct call.

PLANNER

Planner is responsible for choosing and presenting topics. Students may also introduce topics; Planner must notice and respond appropriately when this occurs.

The basis of Planner is a set of pattern-action rules which will be referred to in this paper as *"teaching rules"*. A teaching strategy can be considered to be a set of related teaching rules. Part two of this paper will present a set of teaching rules for the Socratic teaching strategy.

Planner performs its role primarily by searching for, and then applying, an appropriate teaching rule. This rule-selection process starts with a series of policy decisions about various aspects of the presentation. The policy decisions serve both to reduce the set of teaching rules which might apply, and to constrain the application of the rule finally selected.

Teaching rules consist of a condition-part and an action-part, where condition-parts describe situations for which the action-part is well-suited. Condition-parts are represented as the Knowledge State Conditions described in the discussion of Model of the Student. The set of teaching rules applicable at a given point in time consists of those rules whose conditions are satisfied in the current context.

There are three stages of decision-making in rule-selection: ***policy-setting***, ***condition-testing***, and ***conflict-resolution***. (Condition-testing, already discussed, is simply the process of determining which rules are applicable in the current context.)

Policy-setting is the process of making decisions about various aspects of the presentation. In examining tutorial protocols, four critical aspects (or dimensions) of the presentation appear:

(1) ***level*** -- the degree of sophistication;

(2) ***sequence*** -- the order in which concepts are presented;

(3) ***manner*** -- e.g., question-oriented or expository;

(4) ***initiative*** -- who initiates exchanges (tutor, student, or both).

For the remainder of this paper, primarily the first dimension, level of presentation, will be considered. For all dimensions, it will be assumed that general policy decisions (i.e., choices of teaching strategy) are fixed; only specific decisions will be described. [2]

Conflict-resolution is the process of selecting a teaching rule to apply from the set of alternatives (those rules with satisfied conditions). Planner uses several heuristics in this process:

(1) *Policy*: Reject alternatives inconsistent with the current policy decisions.

(2) *Context*: Define the "context" to be the current dependent variable and its immediate factors. Then, if conditions for two different rules are met, but the situation satisfying the conditions for one is more closely related to the context, choose that rule.

(3) *Specificity*: If conditions for two different rules are met, prefer the rule with more specific conditions.

(4) *Blocked Action*: Reject any rules with action-parts that cannot be carried out.

There is also a heuristic for conflicts of a rule with itself:

(5) *Dependent/independent Situations*: If two separate situations satisfy conditions for the same strategy, and one situation is dependent on the other, apply the rule to the independent situation.

In the sample dialogue analysis following, the claim will be made that the Planner's policy decisions, in conjunction with its conflict-resolution heuristics, impose sufficient constraints to determine a unique rule among alternatives.

This control structure is similar in many respects to a production system (Newell, 1973; Newell and McDermott, 1975). A production system consists of a set of data elements (called "Short Term Memory"), a set of "productions", and a supervisor which handles "conflict-resolution". Productions are contingent instructions consisting of conditions paired with actions. The basic cycle of a production system has two steps. In the first, those productions with conditions satisfied by the configuration of elements in Short Term Memory are identified. If several are found they are called a "conflict set"; conflict resolution means choosing a production from this set according to some pre-specified criteria. In the second step, the action part of the selected production is executed. The cycle repeats until the first step fails to find at least one production.

PART II -- A FLEXIBLE SOCRATIC TEACHING STRATEGY

OVERVIEW

This discussion will focus in greater detail on three parts of the system discussed in Part I: the Planner, the Topic Knowledge Representation (TKR), and the Model of the Student (MoS). The goal is to demonstrate how a particular teaching strategy, the Socratic method, could be implemented in the hypothetical dialogue system.

The analogy between Planner and a production system becomes important because of the work on analyzing the Socratic method currently being done by Allan Collins of Bolt, Beranek, and Newman, Inc. (Collins, 1977).

Collins is interested in representing the Socratic method as a set of very high level productions, each of which suggests actions for the tutor to take when some situation occurs. His paper presents 23 production rules for Socratic tutoring.

This section discusses Collins' analysis and shows how his productions could be implemented in an Educational Dialogue System. The system is then demonstrated by tracing through a sample dialogue. In the process, Planner and Topic Knowledge Representation will be worked out in detail, and the Model of the Student will be illustrated informally.

Collins points out that for his productions to constitute a complete specification of the Socratic method requires a higher level theory of teaching strategy be added on, in order to determine which rules are most appropriate to invoke in different situations. Although much remains to be worked out, this paper represents the beginnings of such a theory. Of particular importance are the division of strategies into multiple aspects and the conflict-resolution heuristics, both presented in the preceding Planner discussion and further illustrated later in this section.

SUMMARY OF COLLINS' ANALYSIS

The key idea in the Socratic method, according to Collins, is that of "causal dependencies", the underlying relationships between facts. The method forces students to learn to think for themselves, to induce general principles from knowledge of specific cases, and to apply those general principles to new cases. Thus, three types of things are being taught: 1) specific information; 2) causal dependencies; and 3) reasoning skills -- mainly pertaining to hypothesis generation and testing.

A fairly simple representation of the knowledge to be taught is considered satisfactory for the purposes of his discussion; situations related to each other by one being either necessary or sufficient for the other. The simplicity is made possible by placing primary emphasis on causal dependencies (which are manageable in such a framework), and on reasoning skills (which are taught implicitly by the production rules, and thus need not be represented formally).

Before considering the content of Collins' production rules, it's necessary to

understand the terminology used to state them. An explanation is most easily made in the context of an example.

Imagine that a tutor was trying to teach a student about a mathematical function:

$$f(x) = \begin{cases} x^2 & \text{if } x \geq 0 \\ -x^2 & \text{otherwise.} \end{cases}$$

Then "f(x)" is the dependent variable.

There are two sufficient factors determining the dependent variable "f(x)". They are "x^2" and "$-x^2$". These are sometimes called or-connected factors because one *or* the other is sufficient. Collins would diagram the relationship as in Figure 3a.

Insert Figure 3 about here

It is also possible to have necessary factors. "x^2" and "$-x^2$" each have one necessary factor -- "x^2" requires "$x \geq 0$", and "$-x^2$" requires "$x < 0$". This would be diagrammed as in Figure 3b.

If there were more than one necessary factor, they would be diagrammed similarly to or-connected factors except that the arrows would be labeled with "*and*", and would be called "and-connected".

Several terms describe relationships between factors and the dependent variable. Any path connected by arrows is a chain. For example, "$x \geq 0$", "x^2", and "f(x)", form a chain. A factor between two nodes on a chain is called an intermediate factor: "x^2" is an intermediate factor of "$x \geq 0$" and "f(x)". A factor on a chain, but not an immediate predecessor of the dependent variable is called a prior factor. "x^2" is an immediate predecessor of "f(x)"; "$x \geq 0$" is a prior factor.

A particular case of the dependent variable is a call to consider the factors when some specific information is given. For "f(x)", particular cases consist of specified values of x. Possible cases for f(x) are the set of real numbers: f(4), f(-3), and f(2.7) are all particular cases.

The values of a dependent variable are the results of considering particular cases. The value of "f(x)" is 16 when x=4, -9 when x=-3, and 7.29 when x=2.7.

Relevant factors depend on particular cases. For example, "x^2" is relevant when $x \geq 0$, but not when x is negative. Factors are either primary or secondary, depending on whether they are frequently relevant.

Teaching rules are productions specifying actions to be carried out when their associated conditions are met. Collins' 23 Socratic teaching rules are listed in Table 1. Consider a sample dialogue about "f(x)" to illustrate the first three teaching rules:

Insert Table 1 about here

Tutor: What would f(x) be when x = 4? *(Rule 1 is selected since this starts a session. The particular case of $x = 4$ is selected for the dependent variable $f(x)$, and the student is asked to predict its value.)*

Student: Sixteen.

Tutor: Why 16? *(Rule 2 is selected because the student has asserted a value for the particular case.)*

Student: Because 4 is greater than zero.

Tutor: Why is that important?

("$X \geq 0$" is a prior factor of the dependent variable "$f(x)$". This satisfies the conditions for Rule 3.)

Student: Because when x is greater than zero, f(x) is x^2, and the square of 4 is 16.

Finally, define a teaching strategy to be any coherent set of teaching rules.

TOPIC KNOWLEDGE REPRESENTATION ISSUES

The problem now is mapping between the structures in Topic Knowledge and the entities called for in the conditions of Collins' rules. Tables 2 and 3 display this mapping. Note that the condition entities are described as items to be computed dynamically with respect to the current dependent variable, which may be either a task or an algorithm.

Insert Table 2 about here

Table 2 specifies what instantiations of the condition entities are possible for tasks, along with their identifying features in Topic Knowledge Representation. There are two groups of entities which are associated with each other (e.g., particular cases involving input are always associated with values involving outputs). Table 3 specifies the same kind of information for algorithms. In this case there are two groupings again; the groupings differ in having different types of "values" (outputs vs. properties).

Insert Table 3 about here

A SAMPLE SOCRATIC DIALOGUE

Let us illustrate the ideas presented here by applying them in a tutorial protocol analysis of a hypothetical dialogue between a tutor (T) and a student (S) about the "binary tree" sorting algorithm. [3] The dialog was constructed to maximize the number of teaching rules and conflict-resolution heuristics illustrated, but is based on transcripts of a number of real tutorials conducted by the author. [4] The discussion concentrates on the algorithm's second half, the recursive traversal rule for selecting items from an already-built tree. The tutor's goal is to help the student understand the rule and express it in a computer program.

It is important to note the assumptions implicitly made about student and subject. When is the Socratic method useful? A number of conditions seem relevant. Topics are primarily concerned with relationships between concepts (Collins' "causal dependencies"). Students are already familiar with prerequisite concepts (though perhaps not their inter-relationships). The student is not driven by a goal to acquire a particular knowledge unit (i.e., is content to be led by the tutor). Also, the student has a problem-solving attitude towards questions (making comments such as, "Don't tell me, I want to figure it out myself").

Insert Dialogue Transcript about here

Insert Figures 4 and 5 about here

Insert Boxes 8-12 about here

AN ANALYSIS OF THE TUTORIAL DIALOGUE

By considering selected portions of the preceding dialogue, this section seeks to illustrate three issues:

(1) rules used by the tutor and the method of their selection (Planner);

(2) knowledge required about the subject matter, and its representation (Topic Knowledge Representation);

(3) knowledge required about the student's knowledge state (Model of the Student).

The student's programs and verbal statements will be viewed as hypotheses about necessary or sufficient steps in carrying out a particular task. The purpose of the analysis is to test the extent to which, by utilizing this view, we can explain the behavior of a flexible, intelligent tutor within a framework of systematic rule applications to a specified data structure. [5] Thus, the interaction between the student (S) and the tutor (T) will be examined from the tutor's viewpoint. On S's turns, the analysis is concerned primarily with what information can be added to the Model of the Student at the end of the turn. For T's turns, the focus is on the processes determining the tutor's actions on that turn.

Line 1

Planner:

The tutorial starts at an intermediate level; it assumes that the student is the type for which the Socratic method is appropriate (see above).

Rule 1 ("Known case") is selected, since this is the start of the session. There are two possible actions (see Table 1, Rule 1). The first gives away more information and thus is somewhat easier; at intermediate levels, the easier option is taken. Speed, a motivational factor in picking an algorithm, is selected as the question's topic.

Topic Knowledge Representation:

There are two sorting algorithms: "binary tree" and "straight selection". The binary tree sort is fast; when a quick sorting algorithm is required, it should be used because it has the desired property. (This information is represented in Figure 2.)

Line 2

Topic Knowledge Representation:

The binary tree sort consists of building a tree and traversing it (illustrated in Figure 2).

Model of the Student:

S knows of the tree building algorithm, part of only one known algorithm (which, also, has the property just queried). This indicates S is at least partly aware of the tree sort.

Line 4

Model of the Student:

The student has identified the two necessary factors for the binary tree sort algorithm. This indicates that S knows the answer to the query of Line 1 (without implying that the algorithm is completely understood).

Line 5

Planner:

The intermediate level and failure to completely specify the algorithm at Line 2 suggest that S doesn't completely understand the algorithm, leading to a sequence policy decision establishing the binary tree sort algorithm as the new dependent variable.

"Prior factors" (Table 1, Rule 4) applies because "traversing the tree" is an immediate predecessor factor of the binary tree algorithm (the dependent variable). "Specify functional relationship (Table 1, Rule 11) is also a contender -- S hasn't specified why the binary tree sort is fast -- but it violates the "context" conflict-resolution heuristic because it applies to the old dependent variable.

Topic Knowledge Representation:

The factors of "traversing the tree" are illustrated under the "traverse" node in the shaded part of Figure 6. Traversal involves following the tree, starting with the root. "Following" consists of checking that the node is empty, and (if not) following its left link, then printing it, then following its right link.

Insert Figure 6 about here

Lines 6-7

Model of the Student:

The student is unable to hypothesize the factors relevant to traversing the tree.

Line 8

Planner:

The level of presentation policy, set at Line 3, and the choice of topic policy, set at Line 5, are reinforced by the Model of the Student's identification of a large knowledge gap.

Both "Ask for relevant factors" and "Consider relevant factors" (Table 1, Rules 17 and 23 respectively) are candidates, since S cannot make a prediction. The latter suggest factors while the former assumes the student already knows them; the level policy thus mandates Rule 23.

Topic Knowledge Representation:

A property of the tree built in the binary sort's first part determines the traversal method. (For now, simply call this property, "TREE-PROPERTY".) This connection is provided by a predicate chain linking "build tree" with TREE-PROPERTY and another chain from TREE-PROPERTY to "traverse tree" (the shaded part of Figure 7). Respectively, the two chains assert that: 1) "Build tree" has the property that its ***output*** has TREE-PROPERTY; and 2) "Traverse tree" has the property that its ***input*** has TREE-PROPERTY.

Insert Figure 7 about here

Lines 12-21

Topic Knowledge Representation:

Figure 1 shows the structure network for the tree building algorithm.

Model of the Student:

Comparing Topic Knowledge Representation (Figure 1) to the student's statement shows that S has listed all factors of tree building.

Line 22

Planner:

"Intermediate factors" (Table 1, Rule 3) is the only applicable rule. A motivating factor in the algorithm is that the properties of "build tree's" output facilitate the traversal algorithm. What is needed from S now is the intermediate step between "build tree" and "traverse tree" (that is, the facilitating TREE-PROPERTY).

Topic Knowledge Representation:

The critical TREE-PROPERTY summarizes as:

For any node X,
  for all nodes Y in the tree with root X,
    either: 1) X is the same as Y,
    or 2) Y is less than X and is in X's left sub-tree,
    or 3) Y is greater than X and is in X's right sub-tree.

The representation for TREE-PROPERTY is shown in the unshaded portion of Figure 7 (shaded regions represent connections discussed earlier).

Lines 23-24

Model of the Student:

S is aware of TREE-PROPERTY and that it is shared by "build tree" and "traverse tree" (see Lines 1-4 of the dialogue for further evidence).

Line 25

Planner:

Lines 12-21 and 23-24 indicate that S knows some fairly detailed information, which raises the level of presentation.

"Specify functional relationship" (Table 1, Rule 11) is the only possibility. TREE-PROPERTY, a property of the "traverse tree" algorithm (see Figure 7, shaded region), is also a motivation for its steps (introduced in Figure 6, shaded region). Given a factor for choosing a value (in this case, the algorithm's steps), Rule 11 asks S to state how the factor determines the value. (Note that a value can be a group; as here the question pertains to the set of steps composing the "traverse tree" algorithm.)

Topic Knowledge Representation:

The "tree property" (Figure 7, unshaded region) motivates the traversal algorithm through knowledge about other properties/effects of traversal (Figure 6, unshaded region). Following left links before printing a node causes everything in its left sub-tree to be printed before it. Printing the node before following its right links causes everything in its right sub-tree to be printed after it. Via a chain of HAS-PROPs, "traversal" inherits the property of causing these two situations. This, together with TREE-PROPERTY of Figure 7, causes "traversal" to have its output's property of ascending order.

That this causal relationship exists is a reason to use the set of actions shown for "traversal" in Figure 6 (shaded region) in order to carry out the algorithm.

Lines 26-32

Model of the Student:

S jumped past the motivating factors to the traversal algorithm itself. Comparison of Box 8 to the Topic Knowledge Representation (Figure 6) shows that S hasn't identified all necessary conditions. Particular differences are that S believes it sufficient to consider only sub-trees of the root, and sufficient to only consider one link for nodes in the sub-trees.

Line 33

Planner:

The level of presentation policy is becoming increasingly settled. The effect is that mid-level strategies will be preferred, e.g., suggesting factors to the student rather than asking S to list them or restating factors already given.

There is a problem applying "Counter-example: insufficient factor" (Table 1, Rule 6), since there are two cases of insufficient factors: printing the root and considering all links are both essential. The "dependent/independent situation" conflict heuristic resolves this problem.

For all trees, printing the root will improve the output of S's algorithm in Box 8. Only for some trees does the second factor improve results, since trees similar to Figure 4a won't evoke the second bug. Since considering links is relevant only in situations comprising a sub-set of the criticial situations for root-printing, the latter is the independent condition. Thus, root-printing is chosen as the topic for Rule 6.

"Counter-example: insufficient factor" is chosen only after "Generalize: insufficient factor" and "Demonstrate: missing factor" (Table 1, Rules 5 and 9 respectively) are rejected. S has predicted a value on "traversal". This is a task; its values are possible algorithms (only one of which is under consideration here). S has described an algorithm, though not quite the correct one; that is, S has hypothesized the factors necessary for the algorithm to work. Therefore, any rule dealing with assertions about factors is likely to apply. Rule 6 is preferred over Rule 5, despite their identical conditions, because of the mid-level presentation policy decision described above. Rule 9 is quite similar to Rule 6, but is eliminated by the "specificity" conflic-resolution heuristic.

Carrying out "Counter-example: insufficient factor" (the winning rule) calls for a disconfirming value on the dependent variable. Any input tree suffices; the factors given by S fail to predict the root appearing in the output sequence for the traversal algorithm.

Topic Knowledge Representation:

Output for the tree of Figure 4a can be obtained by executing the correct algorithm (Figure 6) with the tree as input. Obtaining information about processes by simulating them in this manner has become an issue of some recent interest (see Brown, Burton, and Bell, 1974; Brown and Burton, 1975; Scragg, 1975).

Model of the Student:

In conjunction with Topic Knowledge Representation, the Model of the Student can predict the student's algorithm's output by executing the correct algorithm, but considering only the components marked as appearing in the student's version.

Lines 34-38

Model of the Student:

S has now correctly described the output for the algorithm as realized in Box 8.

Lines 40-41

Model of the Student:

The student is now aware of the way in which the proposed algorithm's output was incorrect, and that the missing necessary factor was printing the root between considering left and right links. Thus, S now knows one previously neglected necessary factor of the algorithm, but others remain: S still thinks it sufficient to examine left links only when following the root's left-hand sub-tree, and right links only when in the right-hand sub-tree.

Lines 43-44

Topic Knowledge Representation:

Traversing a binary sort tree correctly requires -- for all nodes -- following their left link, then printing them, then following their right link (Figure 6).

Model of the Student:

S had believed that 1) only left links need be followed for nodes in the left sub-tree, and 2) only right links need be followed for nodes in the right sub-tree.

In these lines (43-44), belief (1) is clearly being modified. It is unclear whether belief (2) has also changed; the conservative approach is to assume not.

Line 45

Planner:

In Lines 43-44, S modified only one of two incorrect assumptions. This drops the presentation level.

As at Line 33, the applicable rules are 5, 6, and 9. S is again ignoring a necessary factors (handling left links of nodes in a tree's right-hand side). Arguments similar to those for Line 33 apply to rule-selection here, except that the lowered presentation level causes selection of the simpler "Generalize: insufficient factor" rule (Table 1, Rule 5).

S's hypothesis appears to be that left sub-tree nodes require following both links, while right nodes require following only right links. S is asked if this is correct.

Line 46

Model of the Student:

Model of the Student's analysis of lines 43-44 is now confirmed. S understands the correct rule for one case, but has not transferred that understanding to the other.

Line 48

Model of the Student:

S finally recognizes the necessity of following both links of all nodes. The current situation, then, is this: S has stated an algorithm (Box 9) which fails in certain cases, awareness of those cases exists, but appropriate fixes aren't yet formulated.

Lines 50-54

Topic Knowledge Representation:

The new algorithm (Box 10) is functionally equivalent to the correct algorithm of Figure 6, since the FOLLOWL and FOLLOWR procedures are identical both to each other and to the last three lines of OUTPUT.

Model of the Student:

The Topic Knowledge Representation analysis above suggests two things: S distinguishes between left-link and right-link nodes (which is unnecessary), and S distinguishes between the root and nodes below it (also unnecessary).

Line 55

Planner:

"Generalize: unnecessary factor" (Table 1, Rule 7) is selected over the one alternative, "Counter-example: unnecessary factor" (Table 1, Rule 8). Both rules can be conditioned solely upon a sufficient factor being thought necessary.

Rule 8, as the higher presentation level rule, would be preferred. Unfortunately, it calls for finding a case with a "wrong" value on the factor and the correct value on the dependent variable. Since the student's version of the algorithm is functionally equivalent to the tutor's, no such value can exist. [6] Thus, the "blocked-action" conflict-resolution heuristic forces rejection of Rule 8 in favor of Rule 7.

Next to be resolved is which unnecessary factor Rule 7 will act on. This is decided by the conflict-resolution heuristic "dependent/independent situations". Reducing the last three lines of OUTPUT in Box 10 to a single line, "FOLLOW(ROOT)",

depends on having reduced FOLLOWL and FOLLOWR to the single procedure FOLLOW. Therefore, eliminating the root/sub-tree distinction depends on first eliminating the left/right sub-tree distinction. This is done in Line 55 simply by asking S if the left/right distinction is necessary.

Lines 56-57

Model of the Student:

S now recognizes that the left/right distinction is not necessary. This eliminates one misbelief, but also confirms another's existence: S distinguishes the root from other nodes. The steps leading to the elimination of this belief, and the conclusion of the dialogue, are more or less a repetition of the pattern just presented.

PART III -- SUMMARY

What has just been presented in Part II constitutes a fairly detailed analysis of a tutorial dialogue. When operating at that level of detail for such an extended discussion, the big picture of the overall system is likely to be lost. Let us step back and review this picture; we will first summarize the dialogue analysis, then step back further to review the system, then step back still further to consider what is involved in actually implementing an Intelligent Educational Dialogue System.

One of the major things illustrated by the analysis is that, given only a simple set of policies and conflict-resolution heuristics, it is possible to make reasonable selections of teaching rules. In only three of the tutor's eight turns was there a unique rule which could be applied. Even in one of those cases, a decision must be made between two alternative forms of the rule (Line 1). Nevertheless, it always seems to be possible to find a unique rule.

The selection process, rather than being pre-determined, is actually fairly flexible. For example, the same rules apply at Line 33 and at Line 45: "Generalize: insufficient factor", "Counter-example: insufficient factor", and "Demonstrate: missing factor" (Table 1, Rules 5, 6, and 9, respectively). At Line 33, Rule 6 was selected, but at Line 45 the system, rather than selecting Rule 6 again, reflected a change in the level of presentation by selecting Rule 5.

Often, a number of decisions have to be made before the action to be taken is finally determined. On Lines 33 and 55, we saw cases where, after a rule had been selected, it was necessary to choose between alternative situations to which it could be applied.

We also saw the system make use of a great deal of knowledge about the topic matter. In Lines 8 through 25, various knowledge about considerations of a known part of the binary sort algorithm which motivate the design of the unknown part is used in leading the student to develop hypotheses about what the unknown part (the traversal algorithm) must look like. In Lines 33 through 56-57, knowledge about the correct form of the traversal algorithm is used to detect and diagnose problems with the student's version.

What can be concluded about the structure of the system from examining the dialogue analysis? Clearly, the key part of the system is the Planner; more particularly, within the Planner, key roles are played by *teaching rules*, which determine actions to take in particular situations, by *conflict-resolution heuristics*, which choose among the teaching rules when more than one applies, and by *policy-setting*, which specifies general constraints that restrict the range of possible actions.

The Planner, although the driving force of the system, is still heavily dependent on services rendered by the other components of the system. This paper has primarily concerned itself only with the Model of the Student and Topic Knowledge Representation. Information about the nature of the student and the nature of the subject matter is essential to a knowledgeable, flexible teaching system. These two components are, in turn, dependent on services rendered by each other (and by other

components not discussed in this paper). For example, the Model of the Student makes use of Topic Knowledge Representation as a basis against which to measure the student's knowledge.

What is needed to actually implement an Intelligent Educational Dialogue System? Much of the work would consist of fleshing out the outline sketched in this paper. The Input/Output Handler, for example, requires implementation of a parser and text generater; numerous schemes for both tasks already exist which are capable of handling an adequate range of natural language. Further, improved versions of the I/O Handler would be relatively easy to insert into the system at a later date, since each component of the system is simply a "black box" from the viewpoint of the other components.

A number of other instances where fleshing out would be required appear in the components highlighted in this paper. Choices between alternative teaching rules are frequently made on the basis of differences in the level of sophistication they demand of a student; the teaching rules must therefore be systematically tagged with information about their level. The condition parts of teaching rules are to be implemented through the mechanism of Knowledge State Conditions, which was presented in Part I's discussion of the Model of the Student. Thus, KSCs (which consist of an information collector, an analysis section, and a weighter) need to be specified for each teaching rule condition part. In addition, general support procedures for specifying, invoking, and running KSCs would need to be implemented; processing of the KSCs, it should be remembered, is a shared responsibility of Planner and Model of the Student. Finally, a thorough database of knowledge about the topic matter needs to be built into Topic Knowledge Representation; this paper has shown some, but not nearly all, of the knowledge that needs to be represented.

There are, in addition, three major technical issues which must be dealt with. They have not been addressed in this paper, as they are well beyond its scope, but work on each is essential to final implementation of a system. The first issue lies in the application of the conflict-resolution heuristics. A systematic procedure needs to be specified for determining which heuristic will be used in choosing between alternatives at a given point.

The second and third issues are, respectively, those of ***control structure*** and ***inter-process communications***. These are closely related. The components described in this paper are processes specialized for particular tasks; each controls many sub-processes. A component does its processing whenever necessary. Thus two components may sometimes operate in parallel, other times serially, and still other times as co-routines. This suggests a particular control structure: a distributed processing network. In such a non-hierarchical control structure, there is no controlling process, but rather co-operation between co-equal processes.

We need to ask how such a control structure would be implemented, and how processes operating in this structure would communicate with each other. The answers, it seems likely, lie in the direction of viewing each component as a processor with its own "operating system". Communications between components would take place by message-passing between these "operating systems", which would determine

the priorities with which messages would receive attention and would take care of interrupting on-going processing when necessary. A great deal of the future work to be done on implementing an Intelligent Educational Dialogue System will consist of specifying these "operating systems" and defining the protocols for passing messages between them.

FOOTNOTES

Acknowledgements: Grateful acknowledgement is made to Donald A. Norman, who provided comments and suggestions at every step of this study. This research was supported by the Advanced Research Projects Agency and the Office of Naval Research of the Department of Defense, and was monitored by ONR under Contract No. N00014-76-C-0628. Final drafts of this report were prepared with the use of facilities provided by the departments of Psychology and Computer Science at Carnegie-Mellon University.

[1] "Build tree" is a portion of the binary tree sort algorithm.

[2] In general policy terms, the Socratic method considered in Part Two is a teaching strategy with presentation characterized by: 1) contingency-driven sequence, 2) question-oriented manner, and 3) tutor-driven initiative. (General and specific issues of level will be discussed later on.) Analysis of how general policy decisions are made must await comparison of alternative teaching strategies. This requires representation of other strategies using the same production paradigm, a long-range goal of work in progress by the author at Carnegie-Mellon University.

[3] This method (Knuth, 1973) orders a list of items by first building a binary tree data representation for the list, then using a recursive tree traversal rule to select items from the tree in their correct order. The key to the algorithm is that the tree can be built to have a property that makes successful traversal quite simple.

[4] Subjects were mostly sophomores at the University of California, San Diego, with less than a year of programming experience in ALGOL. In the majority of the tutorials, students were asked to hand-simulate algorithms, but were not asked to state them as computer programs.

[5] In AI methodology, this is a first test for a proposed system. It is a minimal test; a system's ability to explain intelligent behavior is a necessary -- but not sufficient -- condition for the ability to generate it.

[6] The problem is in applying Collins' rules to the Topic Knowledge Representation for algorithms; a distinction has been created between unparsimonious (as opposed to unnecessary) factors which the rules are not equipped to handle.

REFERENCES

Brown, J. S., & Burton, R.R. Multiple representations of knowledge for tutorial reasoning. In D. Bobrow & A. Collins (eds.), *Representation and understanding.* New York: Academic Press, 1975.

Brown, J. S., Burton, R.R., & Bell, A. *SOPHIE: A sophisticated instructional environment* (technical report #2790). Cambridge, Massachusetts: Bolt, Beranek, and Newman, Inc., 1974.

Collins, A. Processes in acquiring knowledge. *Cognitive Science*, 1977, in press.

Dahl, O.-J., Dijkstra, E.W., & Hoare, C.A.R. *Structured programming.* New York: Academic Press, 1972.

Knuth, D. E. *Sorting and searching.* Menlo Park, California: Addison-Wesley, 1973.

Neches, R. *Intelligent educational dialogue systems: a thesis in Artificial Intelligence / Education technologies.* Unpublished B.A. thesis, University of California, San Diego, 1976.

Newell, A. Production systems: models of control structures. In W. Chase (ed.), *Visual information processing.* New York: Academic Press, 1973.

Newell, A., & McDermott, J. *PSG manual* (Computer Science Department technical report). Pittsburgh, Pennsylvania: Carnegie-Mellon University, Computer Science Department, 1975.

Norman, D. A. Memory, knowledge, and the answering of questions. In R.L. Solso (ed.), *Contemporary issues in cognitive psychology: the Loyola symposium.* Washington, D.C.: Winston (Distributed by Halsted Press, John Wiley and Sons), 1973.

Norman, D. A. Cognitive organization and learning. In S. Dornic & P.M.A. Rabbitt, *Attention and Performance V.* London: Academic Press, 1974.

Norman, D. A., Rumelhart, D. E., & the LNR Research Group. *Explorations in cognition.* San Francisco: W.H. Freeman and Company, 1975.

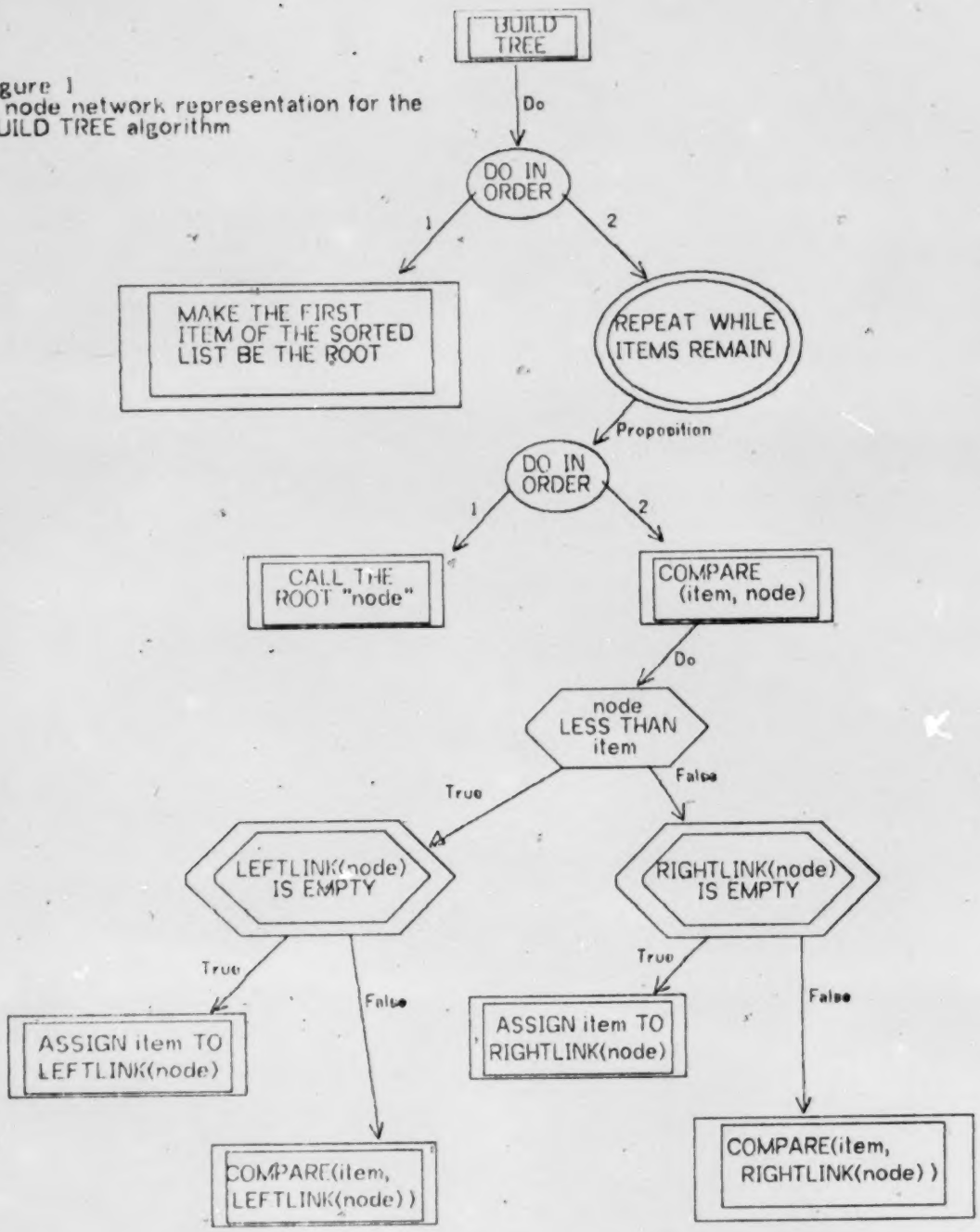
Rothenberg, J. *An intelligent tutor: on-line documentation and help for a military message service* (research report #74-26). Marina del Rey, California: Information Sciences Institute, 1975.

Rumelhart, D. E. Notes on a schema for stories. In D. Bobrow & A. Collins (eds.), *Representation and understanding.* New York: Academic Press, 1975.

Scragg, G. W. Answering questions about processes. In D. A. Norman, D. E. Rumelhart, & the LNR Research Group, *Explorations in cognition.* San Francisco: W.H. Freeman and Company, 1975.

Selfridge, O., & Neisser, U. Pattern recognition by machine. *Scientific American,* 1960, *203*(2), 60-68.

Figure 1
A node network representation for the BUILD TREE algorithm



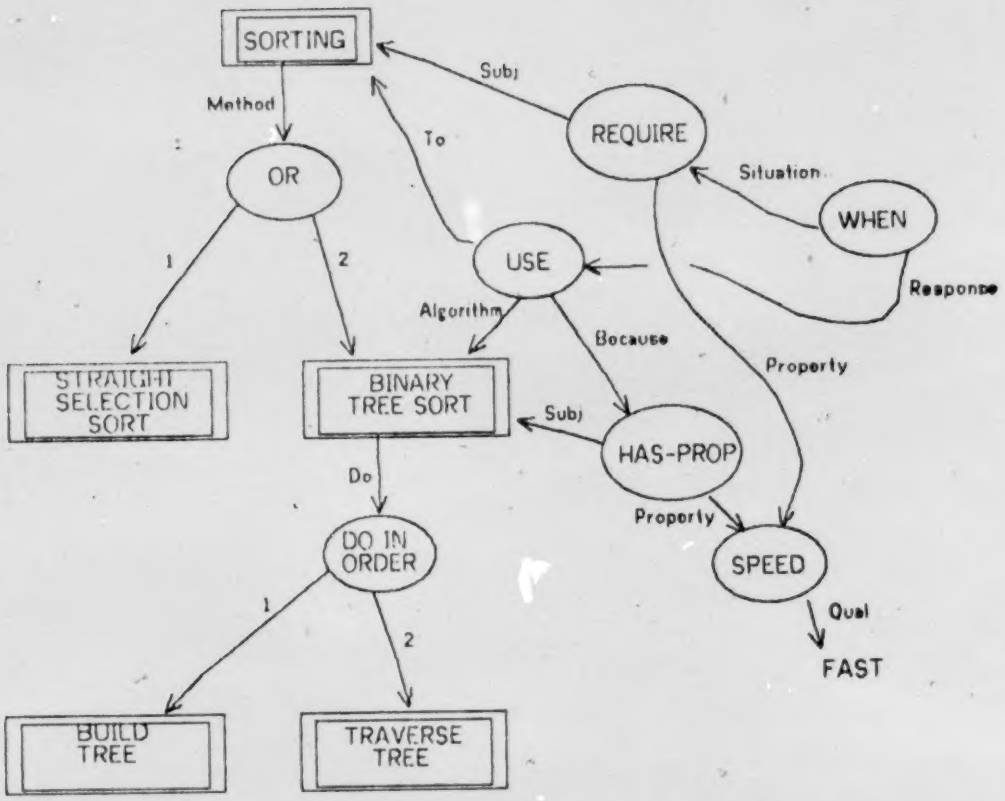


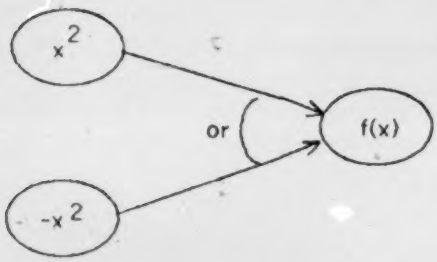
Figure 2
The Topic Knowledge Representation node network for the following statements:

(1) There are two ways to SORT, the STRAIGHT SELECTION SORT and the BINARY TREE SORT.

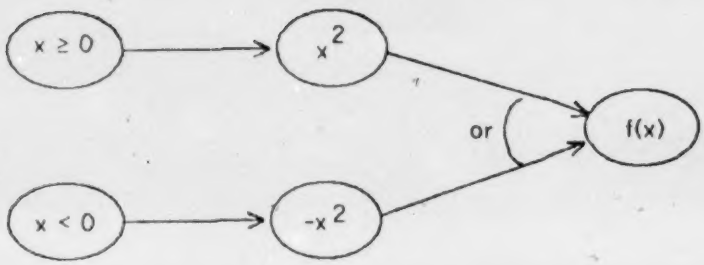
(2) The BINARY TREE SORT is a fast algorithm.

(3) The BINARY TREE SORT consists of two tasks: first build a tree, then traverse it.

(4) When sorting requires speed, use BINARY TREE SORT because it's fast.

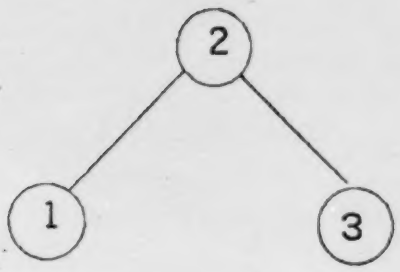


3a. Sufficient factors for determining f(x).



3b. Necessary factors determining the immediate factors of f(x).

Figure 3
Relationships between factors of the function f(x).



4a. A very simple binary sort tree.

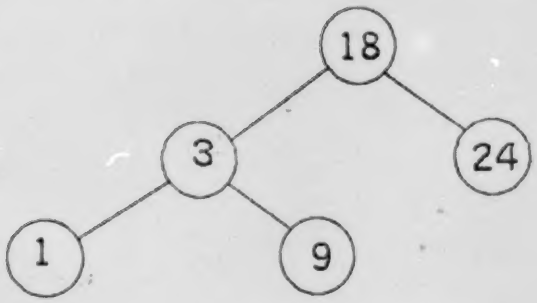
1, 3

4b. The output sequence obtained by applying the program of Box 8 to the tree.

1, 2, 3

4c. The correct output sequence for the tree.

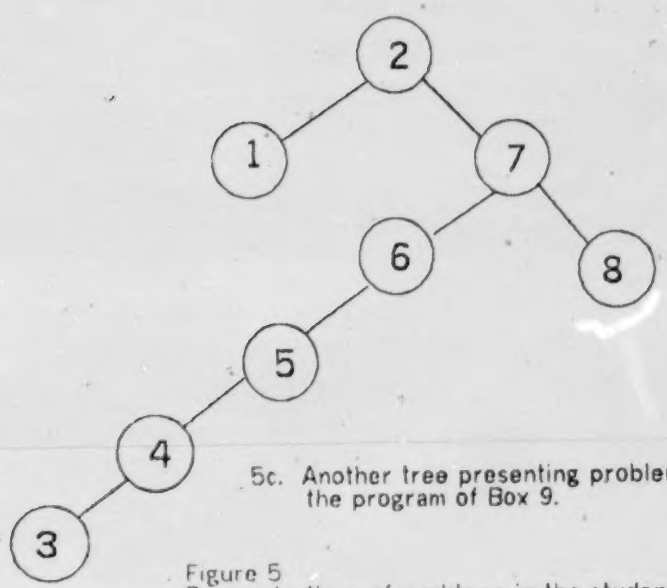
Figure 4
A simple example binary sort tree, used to demonstrate a flaw in the student's statement of the algorithm. The flaw can be seen by comparing the student 's results to the correct results.



5a. A tree presenting problems for the program of Box 9.

1, 3, 18, 24

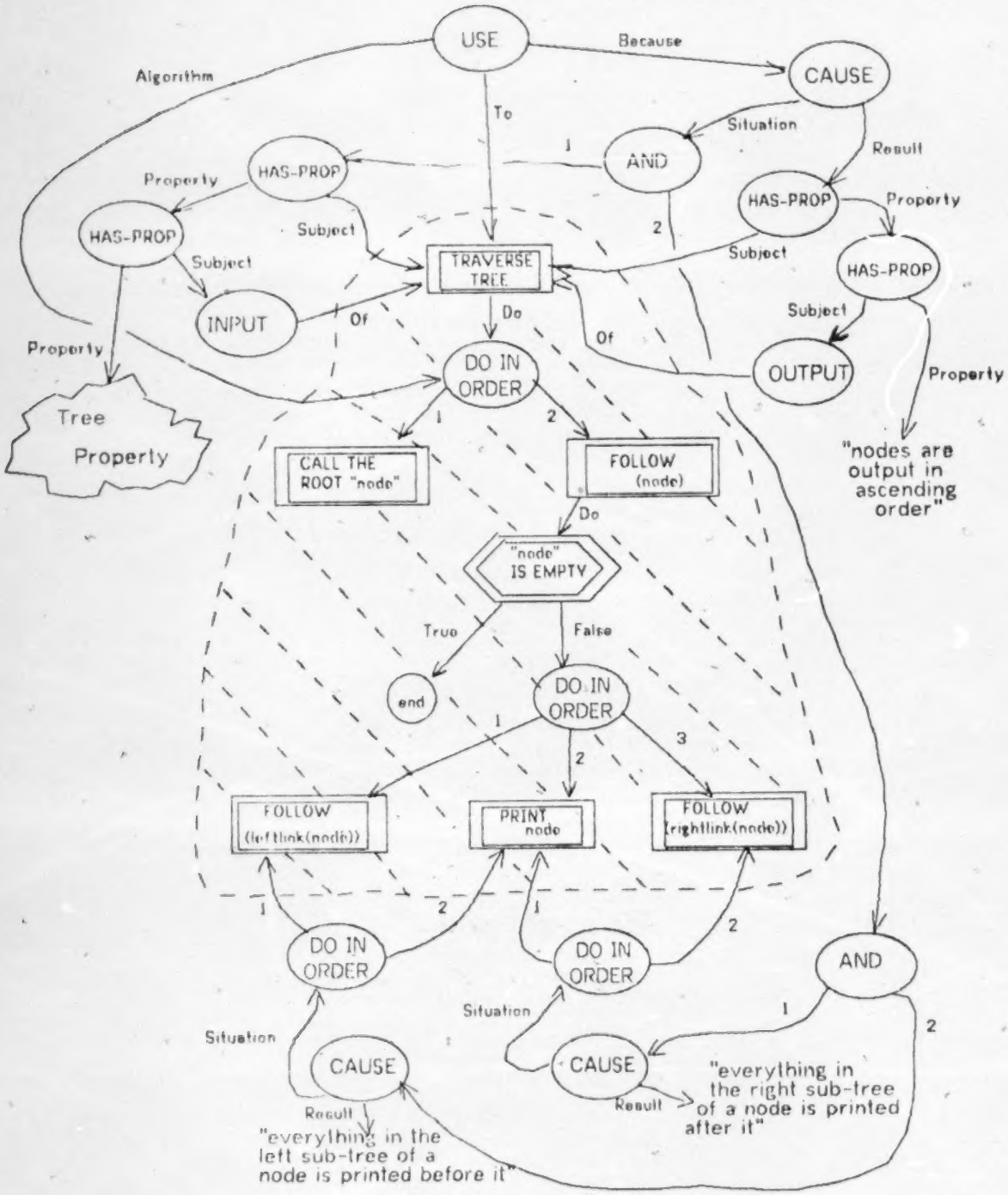
5b. Output sequence for the program of Box 9 on the above tree. (Note that the number "9" was missed.)



5c. Another tree presenting problems for the program of Box 9.

Figure 5
Demonstrations of problems in the student's restated version of the algorithm.

Figure 6
TKR representation for properties and motivations of TRAVERSE TREE



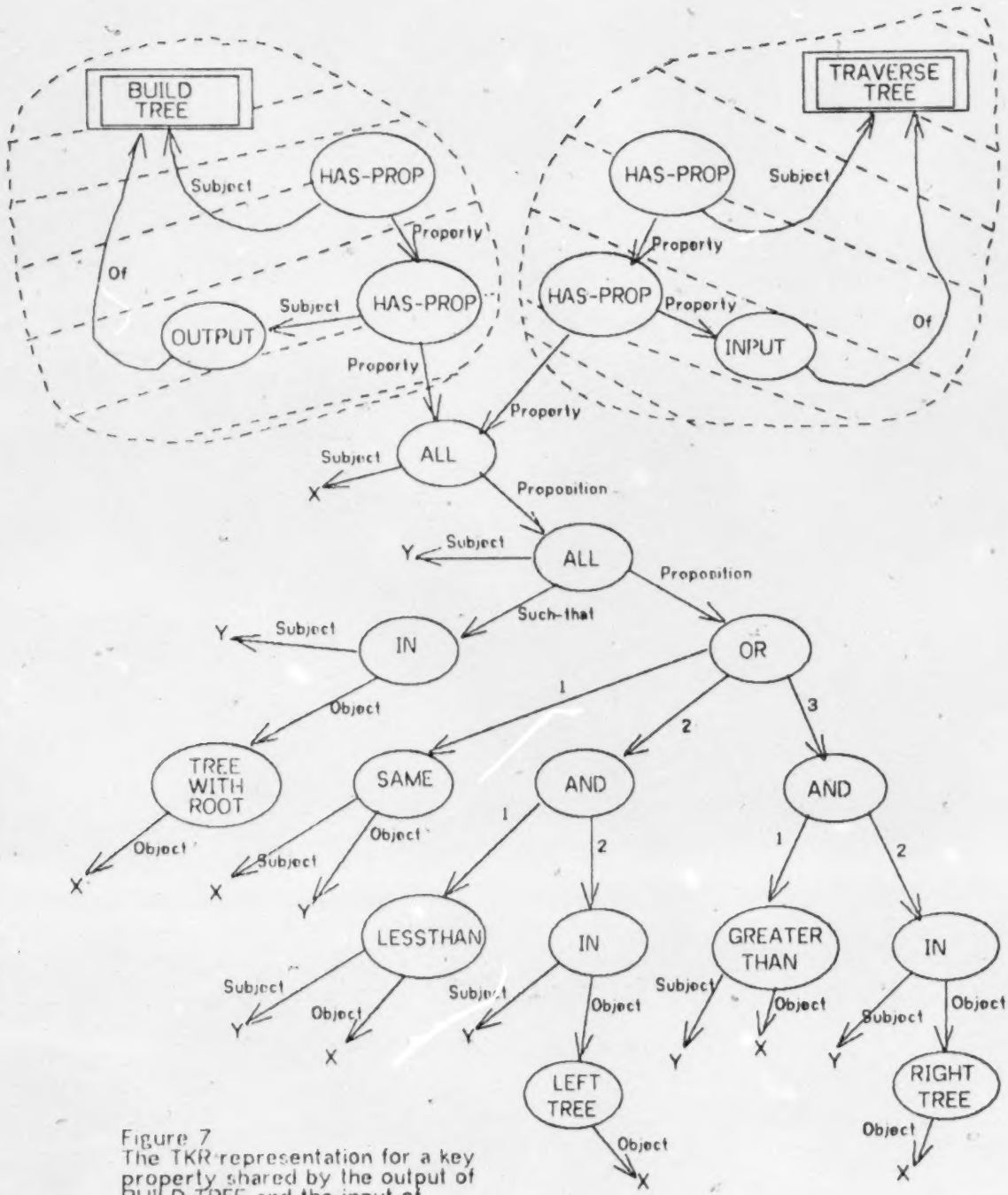


Figure 7
The TKR representation for a key property shared by the output of BUILD TREE and the input of TRAVERSE TREE.

Table 2
Structures needed for applying Collins rules,
and their identifying characteristics in TKR.
(for ALGORITHMS)

	GROUP I	GROUP II
PARTICULAR CASES	Description: Inputs [e.g., a list of numbers to sort]	Description: Properties [e.g., speed]
	Identifying features in TKR: Token of "INPUT" with "SUBJ" relation to the input and "TO" relation to the task node.	Identifying features in TKR: Tokens of "HAS-PROP" with two arguments: a property, and either a task node or a descendent of a task node.
VALUES	Description: Outputs [e.g., a sorted list of numbers]	Description: Algorithms.
	Identifying features in TKR: Token of "OUTPUT" with "SUBJ" relation to the output and "OF" relation to the task node. [Can also be generated and then added to TKR in this format.]	Identifying features in TKR: [See "Factors" under Group I.]
FACTORS	Description: Algorithms [any one of which is sufficient].	Description: Motivations.
	Identifying features in TKR: Node is pointed to by a "METHOD" relation from the structure node for the task.	Identifying features in TKR: Predicate is one of those used in the modified story grammar.

Table 3

Structures needed for applying Collins rules, and their identifying characteristics in TKR.

(for ALGORITHMS)

	GROUP I	GROUP II
PARTICULAR CASES	Description: Inputs.	Description: Inputs.
	Identifying features in TKR: [See "Particular Cases" for Group I in Table 2.]	Identifying features in TKR: [See Table 2.]
VALUES	Description: Outputs.	Description: Properties.
	Identifying features in TKR: [See "Values" for Group I in Table 2.]	Identifying features in TKR: [See "Particular Cases" for Group II in Table 2.]
FACTORS	Description: Components	Description: Motivations.
	Identifying features in TKR: Any structure node which is a descendent of an algorithm node.	Identifying features in TKR: [See "Factors" for Group II in Table 2.]